SIMPLE FILTERS

Filters are the commands which accept data from standard input manipulate it and write the results to standard output. Filters are the central tools of the UNIX tool kit, and each filter performs a simple function. Some commands use delimiter, pipe (|) or colon (:). Many filters work well with delimited fields, and some simply won't work without them. The piping mechanism allows the standard output of one filter serve as standard input of another. The filters can read data from standard input when used without a filename as argument, and from the file otherwise

The Simple Database

Several UNIX commands are provided for text editing and shell programming. (emp.lst) - each line of this file has six fields separated by five delimiters. The details of an employee are stored in one single line. This text file designed in fixed format and containing a personnel database. There are 15 lines, where each field is separated by the delimiter |.

```
$ cat emp.lst

2233 | a.k.shukla | g.m | sales | 12/12/52 | 6000
9876 | jai sharma | director | production | 12/03/50 | 7000
5678 | sumit chakrobarty | d.g.m. | marketing | 19/04/43 | 6000
2365 | barun sengupta | director | personnel | 11/05/47 | 7800
5423 | n.k.gupta | chairman | admin | 30/08/56 | 5400
1006 | chanchal singhvi | director | sales | 03/09/38 | 6700
6213 | karuna ganguly | g.m. | accounts | 05/06/62 | 6300
1265 | s.n. dasgupta | manager | sales | 12/09/63 | 5600
4290 | jayant choudhury | executive | production | 07/09/50 | 6000
2476 | anil aggarwal | manager | sales | 01/05/59 | 5000
6521 | lalit chowdury | directir | marketing | 26/09/45 | 8200
3212 | shyam saksena | d.g.m. | accounts | 12/12/55 | 6000
3564 | sudhir agarwal | executive | personnel | 06/07/47 | 7500
2345 | j. b. sexena | g.m. | marketing | 12/03/45 | 8000
0110 | v.k.agrawal | g.m.| marketing | 31/12/40 | 9000
```

pr : paginating files

We know that,

```
cat dept.lst

01|accounts|6213
02|progs|5423
03|marketing|6521
04|personnel|2365
```

```
05|production|9876
06|sales|1006
```

pr command adds suitable headers, footers and formatted text. pr adds five lines of margin at the top and bottom. The header shows the date and time of last modification of the file along with the filename and page number.

```
pr dept.lst
```

```
May 06 10:38 1997 dept.lst page 1

01:accounts:6213
02:progs:5423
03:marketing:6521
04:personnel:2365
05:production:9876
06:sales:1006
```

…blank lines…

pr options

The different options for pr command are:

-k prints k (integer) columns
-t to suppress the header and footer
-h to have a header of user's choice
-d double spaces input
-n will number each line and helps in debugging
-on offsets the lines by n spaces and increases left margin of page

```
pr +10 chap01
```

starts printing from page 10

```
pr -l 54 chap01
```

this option sets the page length to 54

head – displaying the beginning of the file

The command displays the top of the file. It displays the first 10 lines of the file, when used without an option.

```
head emp.lst
```

-n to specify a line count
head -n 3 emp.lst

will display the first three lines of the file.

tail – displaying the end of a file

This command displays the end of the file. It displays the last 10 lines of the file, when used without an option.

```
tail emp.lst
```

-n to specify a line count

```
tail -n 3 emp.lst
```

displays the last three lines of the file. We can also address lines from the beginning of the file instead of the end. The +count option allows to do that, where count represents the line number from where the selection should begin.

```
tail +11 emp.lst
```
Will display 11th line onwards

Different options for tail are:

- Monitoring the file growth (-f)
- Extracting bytes rather than lines (-c)

Use tail –f when we are running a program that continuously writes to a file, and we want to see how the file is growing. We have to terminate this command with the interrupt key.

cut – slitting a file vertically

It is used for slitting the file vertically. head -n 5 emp.lst | tee shortlist will select the first five lines of emp.lst and saves it to *shortlist.* We can cut by using -c option with a list of column numbers, delimited by a comma (cutting columns).

```
cut -c 6-22,24-32 shortlist
```

```
cut -c -3,6-22,28-34,55- shortlist
```

The expression 55- indicates column number 55 to end of line. Similarly, -3 is the same as 1-3.

Most files don't contain fixed length lines, so we have to cut fields rather than columns (cutting fields).

-d for the field delimiter
-f for the field list

```
cut -d \ | -f 2,3 shortlist | tee cutlist1
```

will display the second and third columns of *shortlist* and saves the output in *cutlist1*. here | is escaped to prevent it as pipeline character

- To print the remaining fields, we have

```
cut –d \ | -f 1,4- shortlist > cutlist2
```

paste – pasting files

When we cut with *cut*, it can be pasted back with the *paste* command, *vertically* rather than horizontally. We can view two files side by side by pasting them. In the previous topic, cut was used to create the two files cutlist1 and cutlist2 containing two cut-out portions of the same file.

```
paste cutlist1 cutlist2
```

We can specify one or more delimiters with -d

```
paste -d "|" cutlist1 cutlist2
```

Where each field will be separated by the delimiter |. Even though paste uses at least two files for concatenating lines, the data for one file can be supplied through the standard input.

Joining lines (-s)

Let us consider that the file *address book* contains the details of three persons

```
cat addressbook
```

paste -s addressbook   -to print in one single line

paste -s -d "| | \n" addressbook   -are used in a circular manner

sort : ordering a file

Sorting is the ordering of data in ascending or descending sequence. The sort command orders a file and by default, the entire line is sorted

```
sort shortlist
```

This default sorting sequence can be altered by using certain options. We can also sort one or more keys (fileds) or use a different ordering rule.

sort options

The important sort options are:

-tchar	uses delimiter *char* to identify fields
-k n	sorts on nth field
-k m,n	starts sort on mth field and ends sort on nth field
-k m.n	starts sort on nth column of mth field
-u	removes repeated lines
-n	sorts numerically
-r	reverses sort order
-f	folds lowercase to equivalent uppercase
-m list	merges sorted files in list
-c	checks if file is sorted
-o flname	places output in file flname

sort –t“|” –k 2 shortlist

sorts the second field (name)

sort –t”|” –r –k 2 shortlist   or

sort –t”|” –k 2r shortlist

sort order can be revered with this –r option.

sort –t”|” –k 3,3 –k 2,2 shortlist

sorting on secondary key is also possible as shown above.

sort –t”|” –k 5.7,5.8 shortlist

we can also specify a character position with in a field to be the beginning of sort as shown above (sorting on columns).

sort –n numfile

when sort acts on numericals, strange things can happen. When we sort a file containing only numbers, we get a curious result. This can be overridden by –n (numeric) option.

cut –d “|” –f3 emp.lst | sort –u | tee desigx.lst

Removing repeated lines can be possible using –u option as shown above. If we cut out the designation filed from emp.lst, we can pipe it to sort to find out the unique designations that occur in the file.

Other sort options are:

```
sort –o sortedlist –k 3 shortlist

sort –o shortlist shortlist

sort –c shortlist

sort –t "|" –c –k 2 shortlist

sort –m foo1 foo2 foo3
```

uniq command – locate repeated and nonrepeated lines

When we concatenate or merge files, we will face the problem of duplicate entries creeping in. we saw how sort removes them with the –u option. UNIX offers a special tool to handle these lines – the uniq command. Consider a sorted dept.lst that includes repeated lines:

```
cat dept.lst
```

displays all lines with duplicates. Where as,

```
uniq dept.lst
```

simply fetches one copy of each line and writes it to the standard output. Since uniq requires a sorted file as input, the general procedure is to sort a file and pipe its output to uniq. The following pipeline also produces the same output, except that the output is saved in a file:

```
sort  dept.lst | uniq – uniqlist
```

Different uniq options are :

Selecting the nonrepeated lines (-u)

```
cut –d "|" –f3 emp.lst | sort | uniq –u
```

Selecting the duplicate lines (-d)

```
cut –d "|" –f3 emp.lst | sort | uniq –d
```

Counting frequency of occurrence (-c)

```
cut –d "|" –f3 emp.lst | sort | uniq –c
```

tr command – translating characters

The tr filter manipulates the individual characters in a line. It translates characters using one or two compact expressions.

tr options expn1 expn2 standard input

It takes input only from standard input, it doesn't take a filename as argument. By default, it translates each character in expression1 to its mapped counterpart in expression2. The first character in the first expression is replaced with the first character in the second expression, and similarly for the other characters.

```
tr '|/' '~-' < emp.lst | head –n 3

exp1='|/' ; exp2='~-'

tr "$exp1" "$exp2" < emp.lst
```

Changing case of text is possible from lower to upper for first three lines of the file.

```
head –n 3 emp.lst | tr '[a-z]' '[A-Z]'
```

Different **tr options** are:
Deleting charecters (-d)

```
tr –d '|/' < emp.lst | head –n 3
```

Compressing multiple consecutive charecters (-s)

```
tr –s ' ' < emp.lst | head –n 3
```

Complementing values of expression (-c)

```
tr –cd '|/' < emp.lst
```

Using ASCII octal values and escape sequences

```
tr '|' '\012' < emp.lst | head –n 6
```

- Source: Sumitabha Das, "UNIX – Concepts and Applications", 4th edition, Tata McGraw Hill, 2006